MODUL 6
Multi Threading

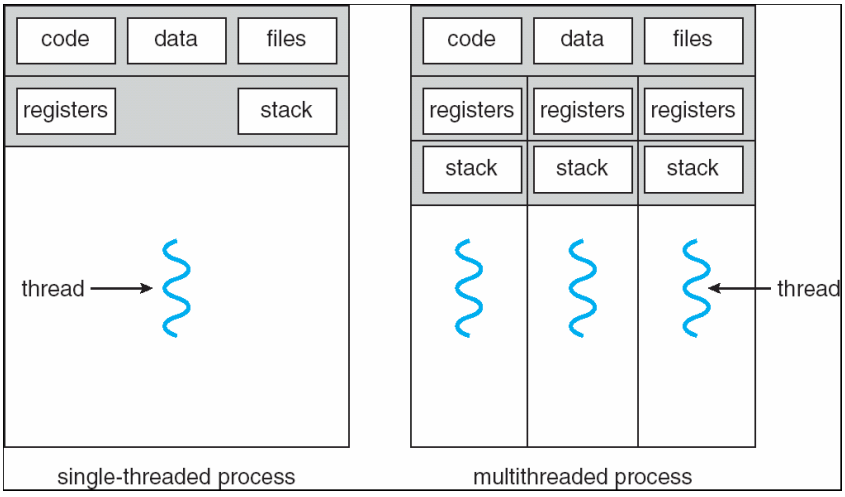
A. Tujuan :

1. Memahami tentang single thread
2. Memahami konsep dan penggunaan dari multi thread

B. Dasar Teori

Pemrograman multithread merupakan konsep penting dalam networking menggunakan Java, misalkan untuk melakukan beberapa tugas yang berbeda dalam satu waktu pada jaringan client-server. Sebelum pembahasan multithread, maka penting untuk memahami perbedaan antar pemrograman single-thread, pemrograman multiprocess, dan pemrograman multi-thread.

Dalam pemrograman single-thread, setiap statemen masing-masing akan dieksekusi satu persatu layaknya sebuah antrian. Pemrograman seperti ini biasa digunakan pada bahasa pemrograman lawas, dimana setiap kode akan dibaca oleh Central Processing Unit (CPU). Pada pemrogaman single-thread ini, apabila suatu statement tidak tereksekusi, maka statement berikutnya juga tidak akan tereksekusi. Dengan kata lain, apabila sebuah variable sedang diakses oleh salah satu statement maka dapat dijamin jika variable itu tidak diakses oleh statement yang lain. Jadi, programmer dapat dengan mudah menentukan statement mana yang mengalami masalah apabila terjadi error pada baris program tersebut.



Gambar 1. Proses single dan multi thread

Berbeda dengan single-thread yang mengeksekusi setiap statement secara berurutan, pada multiproses setiap statement dapat dieksekusi sendiri-sendiri. Sedangkan untuk pemrograman multithread, maka dapat dilihat contohnya pada layar desktop ketika user menjalankan beberapa aplikasi GUI. Berbeda dengan pemrograman multiproses, pada multithread, statement berjalan terpisah namun masih menggunakan

memory yang sama pada komputer, sedangkan pada pemrograman multiproses, memory yang digunakan oleh setiap eksekusi statement akan berbeda.

C. Tugas Pendahuluan

Buatlah desain flowchart untuk setiap soal dalam percobaan

D. Percobaan

D.1. Latihan

1. Pemakaian single thread

```
#!/usr/bin/python

import time

# Define a function
def print_time( proses, delay):
   count = 0
   while count < 5:
      time.sleep(delay)
      count += 1
      print ("%s: %s" % (proses, time.ctime(time.time()) ))

# Create two proses as follows

print_time ("Proses-1", 2 )
print_time ("Proses-2", 4 )
```

Amati hasilnya dan perhatikan bahwa prosesnya dilakukan secara berurutan dari atas ke bawah.

2. Pemakaian multi thread dengan _thread

```
#!/usr/bin/python
import _thread
import time

# Define a function for the thread
def print_time( threadName, delay):
  count = 0
  while count < 5:
    time.sleep(delay)
    count += 1
    print ("%s: %s" % ( threadName, time.ctime(time.time()) ))
# Create two threads as follows
try:
  _thread.start_new_thread( print_time, ("Thread-1", 2, ) )
  _thread.start_new_thread( print_time, ("Thread-2", 4, ) )
except:
  print ("Error: unable to start thread")
while 1:
  pass
```

3. Pemakaian multi thread dengan threading

```
#!/usr/bin/python

import threading
import time

exitFlag = 0

class myThread (threading.Thread):
   def __init__(self, threadID, name, counter):
      threading.Thread.__init__(self)
      self.threadID = threadID
      self.name = name
      self.counter = counter
   def run(self):
      print ("Starting " + self.name)
      print_time(self.name, 5, self.counter)
      print ("Exiting " + self.name)

def print_time(threadName, counter, delay):
   while counter:
      if exitFlag:
         threadName.exit()
      time.sleep(delay)
      print ("%s: %s %s" % (threadName, time.ctime(time.time()),
counter))
      counter -= 1

# Create new threads
thread1 = myThread(1, "Thread-1", 1)
thread2 = myThread(2, "Thread-2", 2)

# Start new Threads
thread1.start()
thread2.start()

print ("Exiting Main Thread")
```

Amati hasilnya, dan bandingkan hasilnya dengan program no 1.

D.2. Permasalahan

1. Buatlah 5 proses pada program Latihan 1 dengan rincian delay sebagai berikut:
   Proses 1: 2
   Proses 2: 3
   Proses 3: 4
   Proses 4: 5
   Proses 5: 6
   Amati proses yang terjadi dan catat hasilnya
2. Buatlah 5 thread pada program Latihan 3 dengan delay 1 dan melakukan **count down** sebagai berikut:
   Thread 1: 3
   Thread 2: 6
   Thread 3: 9
   Thread 4: 12
   Thread 5: 15
3. Buatlah 2 thread dengan delay 1 dan melakukan proses sebagai berikut:
   Thread 1: mencetak bilangan ganjil dari 1..10
   Thread 2: mencetak bilangan genap dari 1..10

E. Laporan Resmi :

1. Analisalah semua program diatas dan buat kesimpulan.